INTERNET DRAFT   Weibin Zhao
October 7, 2005   Henning Schulzrinne
Expires: April 8, 2006   Columbia University

Enabling Global Service Attributes in the Service Location Protocol

draft-zhao-slp-attr-04.txt

Status of This Memo

Copyright Notice

Abstract

This document describes enabling global service attributes in the Service Location Protocol (SLP). A global service attribute describes a service property common to all service types. Its name begins with the "service-" prefix. It is defined via an attribute template, and can be imported into any service template. A single Service Request (SrvRqst) message can use global service attributes to search services across multiple service types. Global service attributes can

be associated with certain special characteristics so as to support advanced discovery scenarios, such as URL changes, multi-access-point services, multi-function devices and replicated services.

1. Introduction

A global service attribute describes a service property common to all service types. For example, transport protocol (such as TCP and UDP) is a global service attribute since it describes a property common to all service types. In contrast, a local service attribute describes a service property specific to certain service type. For example, printing quality (such as draft, normal and high) is a property unique to printers, thus it is a local service attribute.

Currently, all service attributes in the Service Location Protocol (SLP) [RFC2608] are local because they are named, defined and used in the context of a particular service type. Specifically, each service type defines its own attribute set via a service template [RFC2609]; an attribute name is unique only within its service type (i.e., two different service types may use the same attribute name); and an attribute is always used along with its service type. By only using local service attributes, SLP cannot efficiently support certain discovery scenarios. For example, if a User Agent (UA) wants to find all services supporting SCTP (Stream Control Transmission Protocol) [RFC2960], it needs three steps in current SLP: sending a Service Type Request (SrvTypeRqst) message to obtain a list of service types, then using a separate Service Request (SrvRqst) message to search services of each type, and finally combining the search results. As a SrvRqst message can only search services of a single type, n+1 searches are needed for n service types, which is inefficient if n is large.

This document describes enabling global service attributes in SLP, which can improve SLP efficiency and support advanced discovery scenarios.

1.1. Notation Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

2. Enabling Global Service Attributes in SLP

To enable global service attributes in SLP, we need to assign a separate namespace to global service attributes, define them via attribute templates, and using them properly in searching services

across multiple service types.

2.1. Namespace

A global service attribute can be used with any service type. If it has the same name as a local service attribute, then there will be a confusion on which is which. Therefore, a separate namespace is needed for global service attributes. To follow the common practice of prefixing an attribute name with its service type, the "service-" prefix MUST be used in naming global service attributes.

2.2. Definition

A global service attribute is defined using an attribute template. Normally each global service attribute is defined using a separate attribute template; several global service attributes MAY be defined in the same attribute template only if they are to be used together. Any service type that uses a global service attribute imports the attribute's definition into its service template, similar to the C include and Java import mechanisms. In this way, a global service attribute only has one definition, and can be used consistently across all service types.

2.2.1. Attribute Template Syntax

An attribute template is a simplified version of the service type template [RFC2609]. It is defined using the following ABNF [RFC2234]:

```
attr-template = version attr-defs
version       = "# attribute-template-version" version-no term
version-no    = version-no from Section 3.1 of RFC 2609
term          = term from Section 3.1 of RFC 2609
attr-defs     = 1*(attr-def)
attr-def      = attr-def from Section 3.1 of RFC 2609
```

2.2.2. Attribute Template File Name

Similar to the service type template file [RFC2609], an attribute template file has a naming convention defined using the following ABNF.

```
attr-tem-fname = attribute-name "." version-no "." langtag
attribute-name = id from Section 3.1 of RFC 2609
version-no     = version-no from Section 3.1 of RFC 2609
langtag        = langtag from Section 3.1 of RFC 2609
```

The file name of an attribute template is derived from the first attribute name it defines. For example, if a global service attribute

"service-attr-x" is the first attribute in an attribute template, the version number is 1.0, and the language tag is "en", then the attribute template file name is "service-attr-x.1.0.en".

2.2.3. Importing Global Service Attributes

To support importing global service attributes, the ABNF of the service type template defined in RFC 2609 is extended as follows.

```
attr-defs      = *( attr-def / keydef / import-line )
import-line    = "import" attr-tem-fname
attr-tem-fname = attr-tem-fname from Section 2.2.2 of this document
```

2.3. Basic Usages

A global service attribute can appear in any place where a local service attribute is appropriate, such as the attribute predicate in a SrvRqst message, the attribute list in a Service Registration (SrvReg) or Attribute Reply (AttrRply) message, and the attribute tag in a Service Deregistration (SrvDeReg) or Attribute Request (AttrRqst) message.

In a SrvRqst message, when local service attributes are used, exactly one service type MUST be specified; but when only global service attributes are used, multiple service types or a service type wildcard can be specified. A service type wildcard is defined as an empty service type string; the length of the service type string is zero. Therefore, a single SrvRqst message can use global service attributes to search services across multiple or all service types. For example, to find all services supporting SCTP, we can use a single SrvRqst message that has a service type wildcard, and an attribute predicate of "service-transport-protocol=sctp".

2.4. Advantages

Using global service attributes can improve SLP efficiency. First, global service attributes only need to be defined once. Afterwards, they can be imported into any service template. This avoids defining the same attribute repeatedly in different service templates, and ensures a consistent definition. Secondly, by using global service attributes, a single SrvRqst message can search services across multiple service types, which is more efficient than using multiple SrvRqst messages, one for each service type.

3. Advanced Usages

Global service attributes can be associated with certain special characteristics so as to support advanced discovery scenarios. For

example, service identifiers and device identifiers are URIs [RFC2396], such as UUIDs [UUID]. They have two important characteristics: uniqueness and persistence, i.e., each of them uniquely and persistently identifies a service or a device. We can define service identifiers and device identifiers as global service attributes, and use them to support the following discovery scenarios.

3.1. Supporting URL Changes

SLP uses service URLs as service keys, which may be changed (e.g., when a service moves). Therefore, retrieving a service based on its service URLs may not always be feasible, but a UA can always find a service based on its service identifier.

3.2. Discovering Multi-Access-Point Services

A multi-access-point service provides the same service via different access points (i.e., different IP addresses, port numbers or access protocols) residing at the same device. For example, a multi-protocol printer may support IPP [RFC2910] and LPR access protocols, and have two URLs as follows: service:printer:ipp://mpp.example.com and service:printer:lpr://mpp.example.com. A multi-access-point service advertises each access point separately, but all advertisements use the same service identifier and the same device identifier to indicate that they belong to the same service instance residing at the same device. A UA can discover all advertisements of a multi-access-point service by specifying the service identifier and the device identifier as well as the service type (or a service type wildcard) in a SrvRqst message.

3.3. Discovering Multi-Function Devices

A multi-function device provides different types of services at the same device, such as a printing and scanning device. A multi-function device advertises each service type separately, but all advertisements use the same device identifier to indicate that they reside at the same device. A UA can discover all advertisements of a multi-function device by specifying the device identifier and a service type wildcard (or all the service types the device supports) in a SrvRqst message.

3.4. Discovering Replicated Services

A replicated service provides the same service at different devices. It advertises the same service at each device separately, and all advertisements use the same service identifier but different device identifiers. A UA can discover all advertisements of a replicated

service by specifying the service identifier and the service type (or a service type wildcard) in a SrvRqst message. Note that a replicated service uses different device identifiers in its advertisements whereas a multi-access-point service uses the same device identifier in its advertisements.

4. Security Considerations

The security considerations for RFC 2609 apply to this document.

5. Acknowledgments

Erik Guttman's draft on the serviceid: URI scheme [Serviceid] motivated this document directly. Jim Mayer, Mark Bakke and Ira McDonald gave good suggestions. The authors also benefit from the discussions in the SLP mailing list.

6. References

6.1. Normative References

[RFC2608] E. Guttman, C. Perkins, J. Veizades and M. Day, "Service location protocol, version 2", RFC 2608, June 1999.

[RFC2119] S. Bradner, "Key words for use in RFCs to indicate requirement levels", BCP 14, RFC 2119, March 1997.

6.2. Informative References

[Serviceid] E. Guttman, "The serviceid: URI Scheme for Service Location", draft-guttman-svrloc-serviceid-02.txt, August 2002.

[RFC2609] E. Guttman, C. Perkins and J. Kempf, "Service Templates and Service: Schemes", RFC 2609, June, 1999.

[RFC2234] D. Crocker and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[RFC2396] T. Berners-Lee, R. Fielding and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[RFC2910] R. Herriot, S. Butler, P. Moore, R. Turner and J. Wenn, "Internet Printing Protocol/1.1: Encoding and Transport", RFC 2910, September 2000.

[RFC2960] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[UUID] Universal Unique Identifier, http://www.opengroup.org/onlinepubs/9629399/apdxa.htm.

7. Authors' Addresses

Weibin Zhao
Henning Schulzrinne
Department of Computer Science
Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027-7003
Email: {zwb,hgs}@cs.columbia.edu

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.